ABSTRACT

In this paper, we propose two semi-random models for the Minimum Feedback Arc Set Problem and present approximation algorithms for them. In the first model, which we call the Random Edge Flipping model, an instance is generated as follows. We start with an arbitrary acyclic directed graph and then randomly flip its edges (the adversary may later un-flip some of them). In the second model, which we call the Random Backward Edge model, again we start with an arbitrary acyclic graph but now add new random backward edges (the adversary may delete some of them). For the first model, we give an approximation algorithm that finds a solution of cost $(1 + 3\sqrt{3})\text{opt-cost} + n\text{polylog } n$, where opt-cost is the cost of the optimal solution. For the second model, we give an approximation algorithm that finds a solution of cost $O(\text{planted-cost}) + n\text{polylog } n$, where planted-cost is the cost of the planted solution.

Additionally, we present an approximation algorithm for semi-random instances of Minimum Directed Balanced Cut.

Categories and Subject Descriptors
F.2 [Theory of Computation]: Analysis of Algorithms

Keywords
Minimum Feedback Arc Set, semi-random model, average-case analysis, approximation algorithm

1. INTRODUCTION

The Minimum Feedback Arc Set (FAS) problem is one of the most basic optimization problems in the context of directed graphs. The problem asks to find the minimum set of edges (arcs) whose removal makes a given graph a directed acyclic graph (DAG). As any directed acyclic graph can be topologically sorted, this is equivalent to finding an ordering of the vertices, which minimizes the number of backward edges (according to the ordering).

Definition 1.1 Given a directed graph $G(V, E)$ and a vertex ordering $\pi : V \to \{1, \ldots, n\}$, we say that an edge (arc) $(u, v) \in E$ is a forward edge w.r.t $\pi$ if $\pi(u) < \pi(v)$; otherwise, we say that the edge is a backward edge w.r.t. $\pi$.

Definition 1.2 The Minimum Feedback Arc Set problem is as follows. Given a directed graph $G = (V, E)$, find a vertex ordering $\pi : V \to \{1, \ldots, n\}$ that minimizes the number of backward edges w.r.t. $\pi$. The value of the problem equals the number of backward edges w.r.t. the optimal ordering.

The history of the problem dates back to as early as 1957, when Unger [Ung57] considered it in his paper on asynchronous logical feedback networks. The interest in FAS stems from its numerous applications in simplifying the analysis of feedback systems (removing feedback loops) [Ung57, SR61], finding inconsistencies and cyclic dependencies [You63], and ranking based on pairwise information [KS63, ACN08]. It is the complementary problem of the Maximum Acyclic Subgraph [BS90, CMM07, GMR08], which asks to find the largest acyclic graph (w.r.t. the number of edges) in a given digraph.

The problem has been extensively studied in the literature. Karp [Kar72] showed that the problem is NP-hard in 1972, and Kann [Kan92] showed that the problem is APX-hard in 1992. Furthermore, Guruswami, Manokaran, and Raghavendra [GMR08] showed that there is no constant factor approximation for FAS, assuming the Unique Games Conjecture. In their seminal paper, Leighton and Rao [LR99] gave the first polylogarithmic approximation algorithm for the problem with approximation factor $O(\log^2 n)$. This result was later improved by Seymour [Sey95], who used a clever divide and conquer approach to get an approximation factor of $O(\log n \log \log n)$. His technique was generalized by Even, Naor, Rao, and Schieber to a more general family of ordering problems by the method of spreading metrics [ENRS00]. Several researchers studied the special case of tournaments, directed graphs in which every pair of vertices is connected by a directed edge [ACN08, CFR06, KMS07]; Kenyon-Mathieu and Schudy gave a polynomial-time approximation scheme (PTAS) for FAS in tournaments [KMS07]. To summarize, despite our best efforts we only know polylogarithmic approximation algorithms for the problem, and, assuming the Unique Games Conjecture, we cannot get a constant factor approximation for worst-case instances.
However, our primary interest in such optimization problems comes from many different applications they arise in. The instances that we encounter in the real world are not worst-case instances. Besides, instances that are produced in hardness reductions are often contrived, and we are unlikely to encounter them in practice. Also it may be too hard, if not impossible, to design an algorithm that gives a good approximation guarantee for every instance of FAS. This motivates the study of the average-case complexity of the problem.

**Question 1** Can we design algorithms with better provable approximation guarantees for realistic average-case models for the Minimum Feedback Arc Set problem?

Braverman and Mossel [BM08] studied the problem of sorting in the presence of random noise. The problem corresponds to FAS on tournaments with random noise. In their random model, the instance is generated as follows. There is a planted (complete) ordering of the vertices. An edge is added between every pair of vertices: the edge is directed to respect the ordering with probability $\frac{1}{2} + \varepsilon$ and it is directed against the ordering with probability $\frac{1}{2} - \varepsilon$. Braverman and Mossel show that with high probability, we can find the optimal ordering, and the optimal ordering is at most $O(n)$ far from the planted ordering (in $\ell_1$ distance). This result requires that we have compete (but noisy) information; that is, the digraph must be a tournament. The result does not apply to more general instances of FAS.

In this paper, we propose and study two new semi-random models for FAS, which we believe capture many properties of real world instances. We develop new algorithms for semi-random instances of FAS which give $(1 + \delta)$ and constant factor approximations in these two models (see below for the precise statements). In the process, we also develop a constant factor approximations for a natural semi-random model for Minimum Directed Balanced Cut.

Our models are geared towards the two chief applications of FAS: noisy topological sorting and breaking cyclic dependencies. In both of these semi-random models, a semi-random instance can be seen as an acyclic digraph (a FAS instance with cost 0), augmented with random noise.

### 1.1 Semi-random Models

We now describe the two semi-random models for FAS that we study in this paper.

Our first semi-random model is called the Random Edge Flipping model. It models the problem of topological sorting in the presence of a noisy oracle. Imagine there is a unknown ordering of all vertices in a set $V$. We are given a set of comparisons between some pairs of vertices. However, the comparisons are noisy: with some probability $\varepsilon > 0$, the comparison does not agree with the ordering (we assume that the noise is independent for different pairs of vertices). Our goal is to find an ordering which is consistent with as many comparisons as possible. We represent all comparisons by directed edges between vertices. The FAS instance given by the obtained digraph on $V$ defines a semi-random instance in the Random Edge Flipping model.

Let us consider an example in the rank aggregation setting to illustrate why this model seems to capture many real world instances. Suppose that we need to rank web pages (or movies etc.) for a recommendation system. We are given comparisons between some (but not all) of the pairs of web pages, based on the user feedback. It is natural to assume that there is a true underlying order among these web pages, and our goal is to find it. The pairwise comparisons that we see are likely to be noisy perturbations of the true ordering. Note that it is unrealistic to expect that we have comparisons for all pairs of web pages.

This motivates the following formal definition for our first semi-random model:

**Definition 1.3 (Random Edge Flipping Model)**

We are given a set of vertices $V$ of size $n$, and a parameter $\varepsilon$. A semi-random instance of the Random Edge Flipping model is generated as follows.

1. Adversary chooses a directed acyclic graph on $V$, i.e. she chooses a set of edges $E$ so that $G(V, E)$ is acyclic.
2. Every edge in $E$ is flipped (its direction is reversed) with probability $\varepsilon$ independently at random. The set of flipped edges is denoted by $E_R$.
3. The adversary chooses a subset of edges in $E_R$ (possibly empty) and un-flips them (restores the original direction).

Our second semi-random model for FAS is called the Random Backward Edge model. Informally speaking, the instance consists of an adversarial directed acyclic graph and a set of random backward edges, i.e. random edges directed opposite to the ordering. The planted solution to FAS consists of these random backward edges. Our goal is to find a solution whose cost is comparable to the cost of the planted solution.

**Definition 1.4 (Random Backward Edge Model)**

We are given a set of vertices $V$ of size $n$, and a parameter $\varepsilon$. A semi-random instance of the Random Backward Edge model is generated as follows.

1. The adversary chooses an ordering $\pi$ of $V$.
2. Nature adds a random set of backward edges to $G$: for every pair $(u, v)$ with $\pi(u) > \pi(v)$, nature adds an edge $(u, v)$ to $G$ with probability $\varepsilon$. The choices for all edges are independent. The set of random edges is denoted by $E_R$.
3. The adversary adds an arbitrary set of forward edges w.r.t. $\pi$. She also removes some set (possibly empty) of backward edges added by nature.

(Note that this model is stronger than a model where the adversary first adds forward edges, and nature then adds random backward edges.)

This model tries to capture settings where FAS is used to remove cyclic dependencies (or feedback loops) in a system. As an example, imagine we need to schedule a set of jobs on a machine. There are dependencies between jobs described by a digraph: a directed edge from job $i$ to job $j$ indicates that job $j$ needs to be executed before job $j$. A valid schedule is an ordering of these jobs. Ideally, jobs can be ordered in a way that satisfies all dependencies. However, some cyclic dependencies may arise due to some random unlikely events. So the dependency graph consists of

---

1 As an aside, see [Saa77, Saa08] on why pairwise comparisons are often preferred to comparisons between several objects at once.
a digraph (which encodes “genuine” dependencies) and random edges (which encode “accidental” dependencies). Our second model captures such instances.

1.2 Our Results

In this paper, we obtain the following results.

In informal Theorem 1, for every \( \delta > 0 \), there is a polynomial-time algorithm that given a semi-random instance \( G \) of FAS in the Random Edge Flipping model (with noise probability \( \epsilon < 1/4 \)), finds a feedback arc set of cost at most \( 1 + \delta \) opt-cost + \( n \) polylog \( n \) w.h.p., where opt-cost is the cost of the optimal solution in \( G \).

The formal statement of this theorem is given in Theorem 2.2.

Informal Theorem 2 There is a polynomial-time algorithm that given a semi-random instance \( G \) of FAS in the Random Backward Edge model, finds a feedback arc set of cost at most \( O(\text{planted-cost}) + n \) polylog \( n \) w.h.p., where planted-cost is the number of random edges added at step 2 (in other words, it is the cost of the planted solution in \( G \) after step 2).

The formal statement of this theorem is given in Theorem 3.5.

Note that in the first model the approximation additive term is much smaller than the cost of the planted solution if \( \epsilon \Delta \gg \text{polylog } n \) (where \( \Delta \) is the average degree of \( G \)). In the second model, the approximation additive term is much smaller than the number of random backward edges in \( E_R \) if \( \epsilon \gg \text{polylog } n/n \).

There is an important difference between our results for the first and second models. In our result for the first model, we compare the performance of our algorithm with the cost of the optimal solution for FAS. Thus we get an approximation scheme (with an extra additive approximation term) for FAS. In the second model, we compare the performance of our algorithm with the cost of the planted solution. The algorithm gives a true approximation only if the cost of the planted solution is close to the optimal solution (which we believe should be the case for real-world instances).

1.3 Related Work on Semi-random Models

Over the last two decades, there has been extensive research on average-case complexity of many important combinatorial optimization problems. Many algorithms have been proposed for random and semi-random models. In random models, an instance is chosen from a fixed probabilistic distribution of instances (of given size), and the adversary has no influence on the choice. Random models have been studied in the context of graph partitioning problems [BLCS87, DF86, Bop87, JS93, D90, CK99, McS01, CO06], Graph Coloring [AK94, BS95, AKS90], Minimum Steiner Tree [KMSPT86], Densest Subgraph [BCC10] and other problems. On a more related note, Newman [New04] studied random models for Maximum Acyclic Subgraph, the complementary problem of FAS, and showed how semi-definite programming relaxations have integrality gap at most 1.64 for these instances with high probability.

In semi-random models, an instance is generated in a sequence of steps: some of them are random and some are adversarial. Because of that, semi-random instances may have much more structure then completely random instances. Research on semi-random models was initiated by Blum and Spencer [BS95], who introduced and investigated semi-random models for \( k \)-coloring. Later Feige and Kilian [FK98] studied a robust semi-random model for Minimum Bisection, and Feige and Krauthgamer [FK00] studied a semi-random model for Maximum Clique. In all these models, the algorithm can actually exactly find the planted solution.

Most related to our current work are recent results by Kolla, Makarychev and Makarychev [KMM11] on semi-random instances of Unique Games and by Makarychev, Makarychev and Vijayaraghavan [MMV12] on semi-random instances of graph partitioning problems. The algorithms for these models find a solution whose cost is within a constant factor of the cost of randomly added/corrupted edges (as in our second model); in these models, it is not possible to find the planted solution even information theoretically.

1.4 Techniques

The general approach of both our algorithms for semi-random instances of FAS are influenced by [KMM11] and [MMV12]. At a very high level, this approach can be outlined as follows. We write an SDP relaxation\(^2\) for the problem and find a small family of representative SDP solutions. Now for a fixed planted solution, we show that every fixed feasible SDP solution is far from the optimal SDP solution unless it has certain structural properties. Finally, using the union bound and the fact that there are much fewer representative SDP solutions than semi-random instances, we prove that every feasible SDP solution is far from the optimal SDP solution unless it has the required structural properties. Thus the optimal solution has these structural properties w.h.p.

However, techniques from [KMM11] and [MMV12] deal with metric SDP relaxations and are not directly applicable to FAS. In our algorithm for the first model, the Random Edge Flipping model, we introduce a novel SDP relaxation for FAS. Roughly speaking, we start with a very basic LP relaxation for the problem, in which we have an LP variable \( x_{uv} \) for every pair of vertices. Given an ordering \( \pi \) of \( V \), the intended solution corresponding to \( \pi \) is defined by: \( x_{uv} = 1 \) if \( \pi(u) > \pi(v) \), \( x_{uv} = 0 \), otherwise. We require that \( x_{uu} + x_{uv} = 1 \) and \( 0 \leq x_{uv} \leq 1 \). The objective function is to minimize \( \sum_{(u,v) \in E} x_{uv} \). Clearly, this LP is a relaxation. However, it is a very weak relaxation since the optimal value is always equal to 0 — just consider the solution in which \( x_{uv} = 0 \) if there is an edge from \( u \) to \( v \), \( x_{uv} = 1 \) if there is an edge from \( v \) to \( u \), and \( x_{uv} = 1/2 \) otherwise (in this example, we assume that if \( G \) contains \( (u,v) \) then it does not contain \( (v,u) \)). Now we strengthen this LP by SDP constraints. We introduce two SDP variables \( L(u) \) and \( R(u) \) for every vertex \( u \). We require that \( x_{uv} = \langle L(u), R(v) \rangle \), and all vectors \( L(u) \) and \( R(u) \) have length \( \sqrt{\log n} \). We obtain an SDP relaxation. Note that this relaxation is very unusual — it is not even clear right away what the intended values of vectors \( L(u) \) and \( R(u) \) are! We show that for almost all randomly flipped edges \( (u,v) \), \( x_{uv} = \langle L(u), R(v) \rangle \geq 1 - \delta \) in the optimal SDP solution w.h.p. Thus we first remove all edges \( (u,v) \) with \( x_{uv} \geq 1 - \delta \). We obtain an instance with less than opt-cost \((\text{cost} + n \log n \log n) \) backward edges. Then we just run

\(^2\)To obtain some of their results, [KMM11] and [MMV12] use C-SDP or local SDP relaxations.
the “off-the-shelf” algorithm for FAS by Seymour [Sey95] and get a linear ordering.

It is interesting that we get a true approximation algorithm for the Random Edge Flipping model: we compare the cost of the solution with the cost of the optimal solution. That is different from results for Unique Games [KMM11], graph partitioning problems [MMV12] and our second model — all of them compare the cost of the solution with the cost of the planted solution.

Our algorithm for the second model, the Random Backward Edge model, reduces the problem to the Minimum Directed Balanced Cut problem using the reduction of Leighton and Rao [LR99]. Recall that Minimum Directed Balanced Cut asks to cut a directed graph \( G \) into two balanced pieces \( S \) and \( T \) so as to minimize the number of directed edges going from \( S \) to \( T \). In our paper, we present a constant factor approximation algorithm for a natural semi-random model for this problem (see Definition 3.3 for details). Then we use a slightly modified reduction of Leighton and Rao: we find an approximate directed balanced cut \((S, T)\) in the graph, then recursively solve the problem on induced subinstances on \( S \) and \( T \), and finally we combine the obtained solutions.

Since our algorithm for Minimum Directed Balanced Cut gives a constant factor approximation, the cost of the directed cut performed in the first level of recursion is at most \( O(|E_R|) \) (\( |E_R| \) is the number of random edges). We show that the total costs of cuts performed in consecutive levels of recursion go down geometrically (roughly speaking).

Thus the total cost of the obtained solution is \( O(|E_R|) \). In fact, we slightly change the reduction of Leighton and Rao [LR99]: we stop the recursion once a subinstance contains fewer that \( n/(\log n \log \log n) \) vertices and then run the algorithm of Seymour [Sey95]; we do that, roughly speaking, because we pay some small additive penalty every time we find a directed balanced cut. Our algorithm for Minimum Directed Balanced Cut resembles the algorithm for Minimum Balanced Cut (undirected) from [MMV12]. We use the same iterative approach: in iteration \( t \) we solve an SDP for the problem, preprocess the graph (remove “heavy vertices”) and then remove all edges that are “longer” then a threshold value \( \delta = 2^{-t} \). The key difference is that we have to use a directed distance on the graph, and remove edges that are longer than \( \delta \) with respect to this directed distance.

Some properties of directed metrics are quite different from properties of regular undirected metrics. For example, two balls of radius 0 with distinct centers may overlap; because of that it is more difficult to remove non-overlapping balls from a directed metric space. We overcome these difficulties by introducing an additional preprocessing step (which we run in every iteration) that either returns a good directed balanced cut in the graph or finds a large subset of the graph where the directed metric is well approximated with an undirected metric (up to an additive error of \( O(\delta) \)).

Further Applications of Our Techniques. In follow-up research, we showed how to use methods developed in this paper to solve semi-random instances of the Correlation Clustering Problem.

2. RANDOM EDGE FLIPPING MODEL

We set some notation that we use in this section. Denote by \( F^{\pi,G} = \{(u, v) \in E : \pi(u) < \pi(v)\} \) the set of forward edges in the graph \( G \) w.r.t. the ordering \( \pi : V \to \{1, \ldots, n\} \).

Denote by \( B^{\pi,G} = \{(u, v) \in E : \pi(u) > \pi(v)\} \) the set of backward edges. For a set of directed edges \( E' \), we denote the set of reversed edges by reverse\((E') = \{(v, u) : (u, v) \in E'\} \). Finally, let \( D = O(\log n \log \log n) \) be the approximation ratio of Seymour’s algorithm [Sey95] for Minimum Feedback Arc Set.

In the Random Edge Flipping model, the adversary picks an arbitrary directed acyclic graph \( G_0 = (V, E_0) \). Then the nature picks a random subset of edges \( E_R \subset E_0 \): the nature independently adds every edge \((u, v) \in E_0 \) to \( E_R \) with the same probability \( \varepsilon \). Finally, the adversary may replace each edge \((u, v) \in E_R \) with the edge \((v, u) \), remove it, keep it unchanged, or even add an edge \((v, u) \) without removing \((u, v) \). Additionally, the adversary can add an arbitrary set of forward edges. The model is captured in the following definition.

Definition 2.1 Let \( G_0 = (V, E_0) \) be an acyclic directed graph. Let \( \pi \) be an ordering \( V \to [n] \) consistent with \( G_0 \) (that is, an ordering such that all edges of \( G_0 \) are forward edges w.r.t. \( G_0 \)). We define a random set of graphs \( SR(G_0, \varepsilon) \) as follows: Let \( E_R \) be a random subset of \( E_0 \) such that every edge \((u, v) \in E_0 \) belongs to \( E_R \) with probability \( \varepsilon \) (random choices for all edges are independent). Then \( G = (V, E) \in SR(G_0, \varepsilon) \) if for each edge \((u, v) \in E_0 \), \( (u, v) \in E_R \) or \((v, u) \in E_R \), or \((v, u) \in F^{\pi,G} \).

In other words, \( G = (V, E) \in SR(G_0, \varepsilon) \) if \( E_0 \setminus E_R \subset F^{\pi,G} \), and \( B^{\pi,G} \subset reverse(E_R) \).

Note that the definition given above defines a slightly more general model than the one discussed in the introduction. In this model, the adversary may add extra edges which are consistent with the ordering \( \pi \) at the final step.

Theorem 2.2 There exists a polynomial-time approximation algorithm satisfying the following condition. For every directed acyclic graph \( G_0 = (V, E_0) \) on \( n \) vertices with the average degree \( \Delta = 2|E_0|/|V| \), every \( \varepsilon \in (0, 1/4) \), every \( \delta \in (0, 1) \) with probability \( 1-o(1) \) given any graph \( G = (V, E) \) in \( SR(G_0, \varepsilon) \), the algorithm returns an ordering \( \pi : V \to [n] \) of cost at most \((1 + \delta)OPT + O(\varepsilon \log n \log \log n)\), where \( OPT \) is the cost of the optimal solution; the hidden constant in \( O(\cdot) \) may depend on \( \delta \).

Remark 2.3 We can replace the condition \( \varepsilon \in (0, 1/4) \) with \( \varepsilon \in (0, \varepsilon_0) \) for any \( \varepsilon_0 < 1/2 \). We chose \( \varepsilon_0 = 1/4 \) to slightly simplify the proof. However, almost the same proof works for any \( \varepsilon_0 < 1/2 \).

In the analysis, we assume that \( \delta \leq 1/2 \) (otherwise, we may replace \( \delta \) with \( \delta' = 1/2 \).)

SDP Relaxation. We use a novel SDP relaxation for this problem. The SDP is based on the following observation. For every \( n \), there exists a collection of vectors
\[
L^*(1), \ldots, L^*(n) \text{ and } R^*(1), \ldots, R^*(n)
\]
of length \( M = \sqrt{|\log_2 n|} \) such that
\[
(L^*(i), R^*(j)) = \begin{cases} 
0, & \text{if } i \leq j; \\
1, & \text{if } i > j.
\end{cases}
\]
We show that such vectors exist in Lemma 2.5. This lets us use the following intended solution. Every ordering of the vertices \( \pi : V \to [n] \) corresponds to the collection of vectors
removes all edges \((u, v)\) and then runs an "off-the-shelf" algorithm for the Minimum SDP Arc Set Problem by need them in this algorithm.

\[ |B^{\pi,G}| = \sum_{(u,v) \in E} 1(\pi(u) > \pi(v)) = \sum_{(u,v) \in E} \langle L^u(u), R^v(v) \rangle, \]

here \(1(.)\) denotes the indicator function. The vectors \(L^u(u)\) and \(R^v(v)\) satisfy the constraints

\[ ||L^u(u)||^2 = ||R^v(v)||^2 = M^2, \]

\[ \langle L^u(u), R^v(v) \rangle \in \{0, 1\} \tag{1} \]

and

\[ \langle L^u(u), R^v(v) \rangle + \langle L^u(u), R^u(u) \rangle = 1 \]

for every \(u, v \in V\). We relax the constraints (1) and obtain the following SDP.

\[
\min \sum_{(u,v) \in E} \langle L(u), R(v) \rangle \\
\text{subject to: for all } u, v \in V, \\
\langle L(u), R(v) \rangle + \langle L(u), R(u) \rangle = 1 \tag{2} \\
||L^u(u)||^2 = ||R^v(v)||^2 = M^2 \tag{3} \\
\langle L(u), R(v) \rangle \in [0,1] \tag{4}
\]

The variables of the SDP are \(2n\) vectors \(L(u)\) and \(R(v)\), for all \(u, v \in V\). We can also add the triangle inequality constraints \(\langle L(v), R(u) \rangle \leq \langle L(u), R(v) \rangle + \langle L(v), R(w) \rangle\), but we do not need them in this algorithm.

We denote the optimal value of the Minimum Feedback Arc Set Problem by \(OPT\), and the optimal value of the SDP by \(SDP\). If \(\pi\) is the optimal ordering of the vertices, then

\[
OPT = \sum_{(u,v) \in E} 1(\pi(u) > \pi(v)) = \sum_{(u,v) \in E} \langle L^u(u), R^v(v) \rangle \geq SDP.
\]

Hence, \(SDP \leq OPT\).

Algorithm. The algorithm solves the SDP relaxation, removes all edges \((u,v) \in E\) with \(\langle L(u), R(v) \rangle \geq (1-\delta)\), and then runs an “off-the-shelf” algorithm for the Minimum Feedback Arc Set problem.

**Input:** a directed graph \(G = (V, E)\) on \(n\) vertices, parameter \(\delta \in (0,1)\)

**Output:** an ordering of vertices \(\pi_{ALG} : V \rightarrow [n]\)

1. Solve the SDP and obtain a collection of vectors \(L(u), R(u)\) for \(u \in V\).
2. Remove all edges \((u,v) \in E\) with \(\langle L(u), R(v) \rangle \geq (1-\delta)\). Let \(E_+^+\) be the set of the remaining edges, i.e., let
\[
E_+^+ = \{(u,v) \in E : \langle L(u), R(v) \rangle < 1-\delta\}.
\]
3. Run the algorithm of Seymour on the instance \((V, E_+^+)\) and obtain an ordering \(\pi_{ALG} : V \rightarrow [n]\).
4. Return \(\pi_{ALG}\).

Analysis. The proof relies on the following lemma, which implies that the cost of the optimal solution for the graph \((V,E_+^+)\) is small.

**Lemma 2.4** Let \(G_0 = (V, E_0)\) be an acyclic graph on \(n\) vertices with the average degree \(\Delta = 2|E_0|/|V|\), and let \(\pi : V \rightarrow \{1, \ldots, n\}\) be a “planted” ordering such that \(G_0\) does not have backward edges w.r.t. \(\pi\). Then, for every \(\varepsilon \in (0,1)\), every \(\delta \in (0,1)\) and \(D_1, D_2 \geq 1\) satisfying \(D_2 \geq C_5nD_1^2 \log^2 n \log \Delta\) (where \(C_5\) is a parameter depending only on \(\delta\)) with probability \(1 - o(1)\) the set \(SR(G_0, \varepsilon)\) satisfies the following property *: For every \(G = (V, E) \in SR(G_0, \varepsilon)\), for every feasible (not necessarily optimal) SDP solution \(L(u), R(v)\) with SDP value at most \(OPT\),

\[ |B^{\pi,G} \cap E_+^+| \leq (|B^{\pi,G}|/D_1 + D_2), \]

where \(E_+^+ = \{(u,v) \in E : \langle L(u), R(v) \rangle < 1-\delta\}\) is the set of short edges w.r.t. the SDP solution.

We prove the lemma in Section 2.1. Let \(\pi : V \rightarrow \{1, \ldots, n\}\) be a “planted” ordering i.e., an ordering such that all edges in \(G_0\) are forward edges. We show that the algorithm returns a solution of cost at most

\[ (1 + 4\delta)OPT + O_3(n \log^5 n \log \Delta (\log \log n)^3) \]

assuming that the property * holds with

\[
D_1 = \delta^{-1} D = \Omega(\delta^{-1} \log n \log \log n) \quad D_2 = C_5nD_1^2 \log^2 n \log \Delta = O_3(n \log^4 n \log \log n)^2.
\]

Recall, that \(D = O(\log n \log \log n)\) is the approximation ratio of Seymour’s algorithm. We first bound the number of edges removed at the second step. Write,

\[
\sum_{(u,v) \in E} \langle L(u), R(v) \rangle = SDP \leq OPT.
\]

We remove edges \((u,v) \in E\) with \(\langle L(u), R(v) \rangle \geq (1-\delta)\), thus we remove at most \((1-\delta)^{-1}OPT \leq (1 + 2\delta)OPT\) edges.

By our assumption the property * holds which means that the optimal solution for the remaining graph \((V,E_+^+)\) has cost at most \(|B^{\pi,G}+D_2|/D_1 + D_2\). The approximation ratio of Seymour’s algorithm is \(D\), thus, the number of backward edges in \(E_+^+\) with respect to the ordering \(\pi_{ALG}\) returned by the algorithm is at most

\[ D \times \left( \frac{|B^{\pi,G}+D_2|}{D_1} \right) \leq \delta |B^{\pi,G}+D_2| + O(n \log^5 n \log \Delta (\log \log n)^3). \]

The total number of backward edges is at most

\[ (1 + 2\delta)OPT + \delta |B^{\pi,G}+D_2| + O(n \log^5 n \log \Delta (\log \log n)^3). \]

We now show that

\[ |B^{\pi,G}| \leq (1 + 2\delta)OPT + O(n \log^5 n \log \Delta (\log \log n)^3), \]

which concludes the proof. Indeed, let \(L(u), R^v(v)\) be the “integral” SDP solution corresponding to the optimal solution. This solution satisfies the conditions of Lemma 2.4. Now, \(E_+^+\) is the set of forward edges in the optimal solution. Hence, \(|B^{\pi,G}+D_2| \leq |B^{\pi,G} \cap E_+^+| \leq \delta |B^{\pi,G}+D_2| + D_2\), and \(|B^{\pi,G}| \leq (1 - \delta^{-1})(OPT + D_2) \leq (1 + 2\delta)(OPT + D_2)\).

**2.1 Proofs**

**Lemma 2.5** For every natural \(n > 0\), there exists a collection of vectors \(L(1), \ldots, L(n), R(1), \ldots, R(n)\) of length \(\sqrt{\log_2 n}\) such that \(\langle L(i), R(j) \rangle = 1 (i > j)\).
Proof. Without loss of generality we may assume that $n$ is a power of 2: otherwise, we construct $n' = 2^\lceil \log_2 n \rceil$ vectors $L(i)$ and $R(j)$, and then pick the first $n$ vectors. Consider the complete binary tree $T$ of depth $\log_2 n$ with leaves labeled with numbers $1, \ldots, n$ from left to right. The coordinates of the vectors $L(1), \ldots, L(n)$ and $R(1), \ldots, R(n)$ correspond to the internal nodes of the tree. We denote the $x$-th coordinate of $L(i)$ by $L_x(i)$ and the $x$-th coordinate of $R(j)$ by $R_x(j)$. We let $L_x(i) = 1$, if the number $i$ lies in the subtree rooted at the right child of $x$; and $L_x(i) = 0$, otherwise. Similarly, we let $R_x(j) = 1$, if the number $j$ lies in the subtree rooted at the left child of $x$; and $R_x(j) = 0$, otherwise. Let $P_i$ be the path from the root to the leaf $i$. Observe, that $L_x(i) = 0$ and $R_x(i) = 0$ if $x \notin P_i$. Hence, $\|L(i)\|^2 \leq \text{length}(P_i) = \log_2 n$ and $\|R(i)\|^2 \leq \log_2 n$. By definition, $(L(i), R(j)) = \sum_{x \in T} L_x(i) R_x(j)$. Hence, $(L(i), R(j))$ equals the number of $x$‘s such that $i$ is in the right subtree of $x$ and $j$ is in the left subtree of $x$. If $i > j$, then there is exactly one such $x$ — the least common ancestor of $i$ and $j$. If $i \leq j$, then clearly there are no such $x$.

Finally, we pick two vectors $A$ and $B$ orthogonal to all vectors $L(i), R(j)$ and each other (we do this by appending all vectors with two extra coordinates). We add to each $L(i)$ a multiple of $A$ and to each $R(j)$ a multiple of $B$ so that the length of all vectors $L(i)$ and $R(j)$ is exactly $\sqrt{\log_2 n}$. Since $A$ is orthogonal to all $R(j)$’s and $B$ is orthogonal to all $L(i)$’s the inner products $(L(i), R(j))$ do not change. □

Remark 2.6 The bound on the lengths of the vectors in the lemma above is nearly optimal. We can show that the lengths must be at least $\Omega(\sqrt{\log n})$ to satisfy $(L(i), R(j)) = 1(i > j)$. The proof follows from [CMM07]. But, we omit the details here.

We now show that the set of feasible vector solutions to the SDP can be approximated by a much smaller family of representative SDP solutions.

Lemma 2.7 For every graph $G = (V, E)$ on $n$ vertices $|V| = \{1, \ldots, n\}$ with the average degree $\Delta = 2|E|/|V|$, real $M \geq 1$, and $\gamma \in (0, 1)$, there exists a set of matrices $W$ of size at most $|W| \leq \exp(O(\frac{\Delta^2 \log \Delta}{\gamma^2} + n \log n))$ such that: for every collection of vectors $L(1), \ldots, L(n), R(1), \ldots, R(n)$ with $\|L(i)\| = M$, $\|R(j)\| = M$ and $(L(i), R(j)) \leq 0$, there exists $W \in W$ satisfying for every $(i, j) \in E$:

\[
\begin{align*}
    w_{ij} &\leq (L(i), R(j)) \leq w_{ij} + \gamma; \\
    w_{ij} &\in [0, 1].
\end{align*}
\]

Proof. We use the following easy corollary of Grothendieck's inequality: For every set of vectors $L(i), R(j)$ of length $M$, there exists a set of random variables $X_1, \ldots, X_n, Y_1, \ldots, Y_n$ taking values in the set $\{-\tilde{M}, \tilde{M}\}$ (where $M = \sqrt{M^2 M} \leq \sqrt{2M}$, and $\tilde{M}$ is Grothendieck’s constant), such that

\[
\mathbb{E}[X_i Y_j] = (L(i), R(j))
\]

for every $i, j$. We refer to the reader to [AMMN05] (see Lemma 2.3) and [Mak08] (see Theorem 2.8) for more details. Consider $T = 8M^4 \gamma^{-2} \log \Delta$ independent copies of $X_1, \ldots, X_n, Y_1, \ldots, Y_n$. We denote the variables in the $t$-th copy by $X_t(t)$ and $Y_t(t)$. Let $Z_{ij}(t) = X_t(t) Y_t(t)$. Then, $\mathbb{E} Z_{ij}(t) = \mathbb{E}[X_i Y_j] = (L(i), R(j))$ and $Z_{ij}(t) \in \{-\tilde{M}^2, \tilde{M}^2\}$. By the Chernoff bound, for fixed $i$ and $j$,

\[
\Pr \left( \left| \frac{1}{T} \sum_{t=1}^{T} Z_{ij}(t) - (L(i), R(j)) \right| \geq \gamma \right) \leq \exp \left( -\frac{\gamma^2}{2M^2} \right) \leq \exp \left( -\frac{\gamma^2}{2\Delta^2} \right) = \frac{2}{\Delta^2}
\]

Hence, there exists a sample $X^*_t(t), Y^*_t(t)$ such that the condition $\frac{1}{T} \sum_{t=1}^{T} Z_{ij}(t) = (L(i), R(j))$ holds for all but $n$ edges $(i, j) \in E$. Using this observation we define a set of matrices $W_1$:

\[
W_1 = \left\{ \frac{1}{T} \sum_{t=1}^{T} X(t) \otimes Y(t) : X(t), Y(t) \in \{-\tilde{M}, \tilde{M}\}^n \right\}.
\]

Above, $X(t) \otimes Y(t)$ is the rank 1 matrix with entries $(X(t) \otimes Y(t))_{ij} = X_i(t) Y_j(t)$. The set $W_1$ has at most $2^{2\Delta T} n$ elements — this is the number of ways to pick $2T$ vectors $X(t), Y(t) \in \{-\tilde{M}, \tilde{M}\}^n$. As we have just shown using the probabilistic method, for every sequence of vectors $L(i), R(j)$ satisfying the conditions of the lemma, there exists a matrix $W \in W_1$ such that the condition $\|L(i), R(j) - w_{ij} \| \leq \gamma$ holds for all but $n$ edges $(i, j) \in E$. Now we need to truncate all entries of matrices $W \in W_1$ so that they are in the range $[0, 1]$ and shift them so that $w_{ij} \leq (L(i), R(j))$. Let

\[
h(W)_{ij} = \begin{cases} \frac{w_{ij} - \gamma}{\gamma}, & \text{if } w_{ij} - \gamma \in [0, 1]; \\
0, & \text{if } w_{ij} - \gamma < 0; \\
1, & \text{if } w_{ij} - \gamma > 1. \end{cases}
\]

Since $(L(i), R(j)) \in [0, 1]$, if $\|L(i), R(j) - w_{ij} \| \leq \gamma$, then $h(W)_{ij} \leq (L(i), R(j)) \leq h(W)_{ij} + 2\gamma$ (o). Let $W_2 = h(W_1)$. Clearly, $|W_2| \leq |W_1| = 2^{2\Delta T}$.

We are almost done. We only need to take care of $n$ edges for which (o) does not hold. For each matrix $W \in W_2$, we add to the set $W$ the matrix $W$ and all matrices $W'$ that differ from $W$ in at most $n$ positions and whose entries at these positions are multiples of $\gamma$ in the range $[0, 1 - \gamma]$. There are at most $(n^2)^n$ ways to pick at most $n$ positions (for $n > 1$), and there are $[1/\gamma]^n$ ways to pick the new values for them. So for every matrix $W \in W_2$, we add at most $\exp(2n \ln n + n \ln \ln 1/\gamma)$ new matrices to $W$. Then,

\[
|W| \leq 2^{2n T} \cdot 2n \ln n + n \ln [1/\gamma] = 2^{2n T} e^{2n \ln n + n \ln [1/\gamma]} = O((\frac{2n^2 \log \Delta}{\gamma^2}) + n \log n).
\]

□

2.2 Proof of Lemma 2.4

Proof of Lemma 2.4. Let $W$ be the set of matrices for $G_0$ as in Lemma 2.7 with $\gamma = \delta/(2D_1)$ and $M^2 = \log_2 n$. We show that if the property $*$ is violated then there exists a matrix $W$ in the set $W$ satisfying the condition

\[
2 \sum_{(u,v) \in E_R} w_{uv} \geq \sum_{(u,v) \in E_R \setminus E_R} w_{uv} + \delta D_2.
\]

We call such matrix $W$ a witness. Note that $E_R$ constitutes only an $\varepsilon$ fraction of the edge set $E$. We then argue that a fixed matrix $W$ is a witness with exponentially small probability. Using the union bound over all $W \in W$, we conclude that such witness $W \in W$ exists with probability $o(1)$, and, hence, the property $*$ is violated with probability $o(1)$. 


I. Assume that \(*\) is violated. Pick one of the graphs \(G = (V, E) \in \mathcal{S}(G_0, \varepsilon)\) for which the number of backward edges in \(E^+_E\) with respect to the ordering \(\pi\) is greater than \(\left|B\right|/D_1 + D_2\). Denote by \(F = \rho^\pi \subset E\) the set of forward edges with respect to \(\pi\), and by \(B = B^\pi \subset E\) the set of backward edges. Then, \(E_0 \setminus E_R \subset F\) and \(B \subset reverse(E_R)\).

Let \(L(u), R(v)\) be the SDP solution. We know that
\[
SDP \leq OPT \leq |B|,
\]
then
\[
\sum_{(u, v) \in E} \langle L(u), R(v) \rangle = \sum_{(u, v) \in F} \langle L(u), R(v) \rangle + \sum_{(u, v) \in B} \langle L(u), R(v) \rangle \leq \sum_{(u, v) \in B} 1.
\]
Consequently,
\[
\sum_{(u, v) \in B} \langle L(u), R(v) \rangle = \sum_{(v, u) \in B} (1 - \langle L(v), R(u) \rangle) \geq \sum_{(v, u) \in B} \langle L(u), R(v) \rangle.
\]
The first equality follows from the SDP constraint
\[
\langle L(u), R(v) \rangle + \langle L(v), R(u) \rangle = 1.
\]
We now obtain another lower bound on the left hand side. Recall, that we assumed that \(\left|E^+_E \cap B\right| \geq |B|/D_1 + D_2\).

For each \((v, u) \in E^+_E\), we have \(\langle L(v), R(u) \rangle < (1 - \delta)\) (by definition of \(E^+_E\)) and \(\langle L(u), R(v) \rangle \geq \delta\). Hence,
\[
\sum_{(v, u) \in B} \langle L(u), R(v) \rangle \geq \delta |B|/D_1 + D_2.
\]
Therefore,
\[
2 \sum_{(v, u) \in B} \langle L(u), R(v) \rangle \geq \sum_{(v, u) \in B} \langle L(u), R(v) \rangle + \delta (|B|/D_1 + D_2).
\]
Since \(E_0 \setminus E_R \subset F\),
\[
2 \sum_{(v, u) \in B} \langle L(u), R(v) \rangle \geq \sum_{(v, u) \in E_0 \setminus E_R} \langle L(u), R(v) \rangle + \delta (|B|/D_1 + D_2).
\]
Pick \(W \subset W\) such that \(w_{uv} \leq \langle L(u), R(v) \rangle \leq w_{uv} + \gamma\) for \((u, v) \in E_0\). We rewrite the inequality above in terms of \(W\),
\[
2 \sum_{(v, u) \in B} (w_{uv} + \gamma) \geq \sum_{(v, u) \in E_0 \setminus E_R} w_{uv} + \delta (|B|/D_1 + D_2).
\]
Thus,
\[
2 \sum_{(v, u) \in B} w_{uv} \geq \sum_{(v, u) \in E_0 \setminus E_R} w_{uv} + \delta (|B|/D_1 + D_2) - 2\gamma |B|
\]
\[
= \sum_{(v, u) \in E_0 \setminus E_R} w_{uv} + \delta D_2.
\]
Finally, using \(B \subset reverse(E_R)\), we obtain the desired inequality
\[
2 \sum_{(v, u) \in E_R} w_{uv} \geq \sum_{(v, u) \in E_0 \setminus E_R} w_{uv} + \delta D_2.\tag{6}
\]
II. We now estimate the probability (over the choice of the random set \(E_R\)) that a fixed matrix \(W \in W\) is a witness i.e., satisfies (5). Define a random variable \(X_{uv} : X_{uv} = 2\) if \(\langle u, v \rangle \in E_R\) and \(X_{uv} = -1\), otherwise. Then \(EX_{uv} = 2\varepsilon - (1 - \varepsilon) = 3\varepsilon - 1 < -\varepsilon\). Condition (5) can be rewritten as follows
\[
S = \sum_{(u, v) \in E_0} w_{uv} X_{uv} \geq \delta D_2 \equiv \lambda_1.
\]
We denote the left hand side by \(S\), the right hand side by \(\lambda_1\) we also let \(\lambda_2 = \varepsilon \sum_{(u, v) \in E_0} w_{uv}\). We have
\[
E[S] = \sum_{(u, v) \in E_0} w_{uv} E[X_{uv}] \leq -\varepsilon \sum_{(u, v) \in E_0} w_{uv} = -\lambda_2.
\]
and
\[
Var[S] = \sum_{(u, v) \in E_0} w_{uv}^2 Var[X_{uv}] = 9\varepsilon (1 - \varepsilon) \sum_{(u, v) \in E_0} w_{uv}^2 \\
\leq 9\varepsilon \sum_{(u, v) \in E_0} w_{uv} = 9\lambda_2.
\]
By the Bernstein inequality,
\[
\begin{align*}
\Pr\left(S \geq \lambda_1\right) &= \Pr\left(S - ES \geq \lambda_1 + \lambda_2\right) \\
&\leq \exp\left(-\frac{(\lambda_1 + \lambda_2)^2}{2 Var[S] + 2(\lambda_1 + \lambda_2)}\right) \\
&\leq \exp(-C(\lambda_1 + \lambda_2)) \leq \exp(-C\delta D_2),
\end{align*}
\]
for \(C = 1/20\). By Lemma 2.7,
\[
|W| \leq \exp\left(O\left(\frac{nM^4 \log \Delta}{2\gamma^2} + n \log n\right)\right) \\
= \exp\left(O\left(\frac{n D_1^2 \log^2 n \log \Delta}{\varepsilon^2}\right)\right).
\]
Hence, by the union bound (since \(D_2 \geq C_3 n D_1^2 \log \log \Delta\), there exists a witness \(W \in W\) with probability at most \(o(\varepsilon^{-n})\). 

\[\]
3. RANDOM BACKWARD EDGE MODEL

3.1 Minimum Feedback Arc Set

We reduce the Minimum Feedback Arc Set problem to the Minimum Directed Balanced Cut problem. Recall that in the Minimum Directed Balanced Cut problem, we are given a directed graph \(G\) and our goal is to find a balanced cut \((S, T)\) in \(G\) so as to minimize the number of edges going from \(S\) to \(T\). The reduction, which was introduced by Leighton and Rao [LR99] for worst-case instances of FAS, works as follows. Consider an instance \(G\) of FAS. Let \(\pi : V \rightarrow [n]\) be an optimal solution for FAS. Let \(OPT\) be its cost. Note that there is a directed bisection cut in \(G\) of cost at most \(OPT\). Specifically, let \(S = \{u : \pi(u) > n/2\}\) and \(T = \{u : \pi(u) \leq n/2\}\). Then every edge from \(S\) to \(T\) is a backward edge (w.r.t. \(\pi\)); thus, the total number of edges from \(S\) to \(T\) is at most \(OPT\).

The reduction approximately finds a directed balanced cut \((S', T')\) in \(G\), then recursively solves subinstances of FAS on the induced graphs \(G[S']\) and \(G[T']\). Finally, it concatenates solutions for \(G[T']\) and \(G[S']\) (so that every vertex in \(T'\) lies to the left of every vertex in \(S'\)). Note that, for any one level of the recursion tree defined by this procedure, \(OPT\) is an upper bound on the total sum of the optimal solution costs.
of all the Directed Balanced Cut problems on it. Hence, this reduction gives an \(O(\alpha_{MDBC} \log n)\) approximation algorithm for worst case instances of FAS, where \(\alpha_{MDBC}\) is the approximation factor for Minimum Directed Balanced Cut, since the depth of the recursion is \(O(\log n)\), and the cost of all cuts we perform at one level of recursion is at most \(\alpha_{MDBC} \cdot OPT\). If we use the SDP algorithm of Agarwal, Charikar, Makarychev, and Makarychev [ACMM05], then \(\alpha_{MDBC} = O(\sqrt{\log n})\); if we use the LP algorithm of Leighton and Rao [LR99], then \(\alpha_{MDBC} = O(\log n)\).

In this paper, we present a constant factor approximation algorithm for semi-random instances of the Minimum Directed Balanced Cut problem. Then we adapt the reduction to work with this algorithm.

**Definition 3.1** We say that \((S, T)\) is a partition of \(V\) if \(V = S \cup T\) and \(S \cap T = \emptyset\). A partition \((S, T)\) is \(b\)-balanced if \(|S| \geq b|V|\), \(|T| \geq b|V|\) (where \(b \in [0, 1/2]\)). A partition \((S, T)\) is a bisection of \(V\) if \(|S| - |T|\) \(\leq 1\). We say that a cut \((S, T)\) in a graph \(G = (V, E)\) is \(b\)-balanced if \((S, T)\) is a \(b\)-balanced partition of \(V\); similarly, a cut \((S, T)\) is a bisection cut if \((S, T)\) is a bisection partition of \(V\).

We say that a directed edge \((u, v)\) is cut by a directed cut \((S, T)\) if \(u \in S\) and \(t \in T\) (note that if \(u \in T\) and \(v \in S\) then the edge is not cut). The cost of a cut \((S, T)\) in a directed graph \(G\) is the number of edges in \(G\) cut by \((S, T)\). We denote the cost of \((S, T)\) by \(\text{cost}(S, T)\). Given a set of edges \(E\), we denote the number of edges in \(E\) cut by \((S, T)\) by \(\text{cost}_E(S, T)\).

**Definition 3.2** Let \(V\) be a set of vertices and \(\varepsilon > 0\). Construct a random set \(E_R\) of directed edges as follows. Choose every pair \((u, v)\) \(\in V \times V\) independently with probability \(\varepsilon\) and add it to \(E_R\). We say that \(E_R\) is an \(\varepsilon\)-random set of edges.

Our algorithm proceeds by considering semi-random instances of Minimum Directed Balanced Cut, defined recursively based on the partitioning in the previous steps. This motivates the definition of the following semi-random family of graphs.

**Definition 3.3** Let \(V\) be a set of vertices and \(\varepsilon > 0\). Let \(E_R\) be an \(\varepsilon\)-random set of edges. Define the random family of graphs \(\text{SRD}(V, E_R)\) by \(\text{SRD}(V, E_R) = \{H = (U, E_H) : U \subseteq V, (S, T)\) is a bisection of \(U\) and \(E_H \cap (S \times T) \subseteq E_R\}\}.

In other words, graphs \(\text{SRD}(V, E_R)\) are those graphs \(H\) that can be obtained as follows. Choose a subset \(U \subseteq V\) and a bisection \((S, T)\) of \(U\). Then choose a set of edges \(E_1 \subseteq E_R \cap (S \times T)\) and a set of edges \(E_2 \subseteq (S \times S) \cup (T \times T)\). Finally, let \(H = (U, E_1 \cup E_2)\).

We call graphs in the family \(\text{SRD}(V, E_R)\) semi-random directed graphs.

We present an approximation algorithm for the Minimum Directed Balanced Cut problem in semi-random directed graphs in the following theorem.

**Theorem 3.4** There is a randomized polynomial-time algorithm for the Minimum Directed Balanced Cut problem that does the following. Let \(V\) be a set of vertices, \(\varepsilon > 0\), and \(E_R\) be an \(\varepsilon\)-random set of edges. With probability at least \(1 - e^{-cn}\) over \(E_R\), for every graph \(H = (U, E_R) \in \text{SRD}(V, E_R)\), the algorithm given \(H\) finds a \(b\)-balanced cut \((S', T')\) of \(H\) such that

\[
\text{cost}(S', T') \leq C \max(\varepsilon|U|^2, g(n)),
\]

where \(g(n) = n\sqrt{\log n} \log \log n\), and \(c < 0\), \(C > 0\), and \(b > 0\) are some absolute constants.

Proof. We will prove this theorem in the next subsection. We show how to get a constant factor approximation for semi-random instances of FAS using this algorithm. While we follow essentially the same reduction as in the worst-case, from FAS to Minimum Directed Balanced Cut (except that we stop recursing when the instance size is roughly \(O(n/\log n)\)), we only lose a constant factor in the approximation guarantee for semi-random instances, because the total cost we incur in each level of the recursion goes down geometrically.

**Theorem 3.5** There is a randomized polynomial-time algorithm for semi-random instances of the Minimum Feedback Arc Set problem that does the following. Let \(V\) be a set of vertices, \(\varepsilon > 0\), and \(E_R\) be an \(\varepsilon\)-random set of edges. Then with probability \(1 - o(1)\) over \(E_R\), for every vertex ordering \(\pi\) and every directed graph \(G\) in which all backward edges w.r.t. \(\pi\) belong to \(E_R\), the algorithm given \(G\) finds an ordering \(\pi'\) such that \(G\) has at most \(O(\varepsilon n^2 + (\log n)^{3/2} (\log \log n)^{3/2})\) backward edges w.r.t. \(\pi'\).

**Proof.** We may assume that \(\varepsilon > (\log n)^{3/2} (\log \log n)^{3/2}\); as otherwise we let \(\varepsilon' = (\log n)^{3/2} (\log \log n)^{3/2}\) and let \(E_{R}' \supseteq E_R\). We use the recursive approach that we outlined above.

Let \(\pi\) be the planted solution for FAS on \(G\). First, we observe that for every \(U \subseteq V\), \(G[U] \in \text{SRD}(V, E_R)\). Indeed, let \(T\) be the set of the first \(\lceil |U|/2 \rceil\) elements of \(U\) w.r.t. \(\pi\) and \(S\) be the set of the last \(\lceil |U|/2 \rceil\) elements of \(U\) w.r.t. \(\pi\). Note that all edges from \(S\) to \(T\) are backward edges in the planted solution and therefore they belong to \(E_{R}\). Hence \(G[U] \in \text{SRD}(V, E_R)\) and we can apply Theorem 3.4 to \(G[U]\).

We write a recursive procedure that given a set \(U \subseteq V\) finds a linear ordering of vertices in \(U\).

1. If \(|U| \leq n/(\log n \log \log n)\) the algorithm applies the algorithm of Seymour [Sey95] to the induced graph \(G[U]\) and returns the obtained ordering of \(U\).
2. If \(|U| > n/(\log n \log \log n)\), the algorithm finds an approximate directed balanced cut \((S', T')\) using the algorithm from Theorem 3.4. Then it recursively finds linear orderings \(\pi_{T'}\) for \(T'\) and \(\pi_{S'}\) for \(S'\). It returns the concatenation of orderings \(\pi_{T'}\) and \(\pi_{S'}\).

Let us bound the cost of the solution returned by the algorithm. Consider its recursion tree. We charge every backward edge \((u, v)\) to the node of the recursion tree where we separate \(u\) and \(v\).

First consider leaf nodes of the recursion tree. Let us prove that with probability \(1 - e^{-\Omega(n/\sqrt{\log n})}\), for every subset \(U\) of size \(\Theta(n/(\log n \log \log n))\) the induced graph \(G[U]\) contains at most \(2\varepsilon|U|^2\) edges from \(E_{R}\). There are \(\Theta(n/(\log n))\) sets \(U\) of size \(\Theta(n/(\log n \log \log n))\). For each \(U\), the induced graph \(G[U]\) contains at most \(\varepsilon|U|^2\) edges from \(E_{R}\) in expectation. Thus by Chernoff’s bound, \(G[U]\) contains at most \(2\varepsilon|U|^2\) edges from \(E_{R}\) with probability \(1 - e^{-\Omega(\varepsilon|U|^2)} = 1 - e^{-\Omega(n/\sqrt{\log n})}\). By the union bound, we get that with
probability 1 − e−O(\sqrt{\log n}) all subgraphs \( G[U] \) contain at most \( 2e[U]^2 \) random edges.

Since the set \( U \) for every leaf of the recursion tree has size \( |U| = \Theta(n/(\log n \log \log n)) \), we get that \( G[U] \) contains at most \( 2e[U]^2 = O\left(\frac{\varepsilon n}{\log n \log \log n}\right) \) edges (w.h.p. for all leaf nodes). Therefore, the algorithm of Seymour finds a solution of cost at most \( O(\varepsilon n) \cdot |U| \) in \( G[U] \). So the total cost charged to all leaf nodes is at most \( O(\varepsilon n) \cdot \sum |U| = O(\varepsilon n^2) \).

Now consider internal nodes of the recursion tree. An internal node pays for edges from \( S' \) to \( T' \) in \( G[U] \). By Theorem 4.4, the total number of such edges is at most \( C \max(e[U]^2, g(n)) \leq C(e[U]^2 + g(n)) \). Since sizes of sets \( U \) decrease geometrically in the recursion, the sum of \( C\varepsilon n^2 \) over all internal nodes of the recursion tree is at most \( O(\varepsilon n^2) \).

The sum of all terms \( g(n) \) is at most \( g(n) \cdot O(\log n \log \log n) \) since there are \( O(\log n \log \log n) \) nodes in the recursion tree.

We get that w.h.p. the cost of the solution is

\[
O(e[U]^2 + g(n) \log \log n).
\]

\( \square \)

4. PROOF OF THEOREM 3.4

4.1 Proof Overview

Our algorithm is similar to the algorithm for undirected Minimum Balanced Cut from [MMV12]. It iteratively removes edges and vertices from the graph \( H \). It starts with \( H_0 = H \). Then it iteratively constructs a sequence of subgraphs \( H_0 \supset H_1 \supset H_2 \supset \ldots \). In iteration \( t \), the algorithm solves an SDP relaxation for the Minimum Directed Balanced Cut problem in \( H_t \), performs some preprocessing by removing some vertices from \( H_t \) and then cuts some edges in the remaining graph. In the obtained graph \( H_{t+1} \), the number of edges from \( E_R \) drops by a constant factor (compared to \( H_t \)). After at most \( O(\log \log n) \) rounds, the algorithm gets a graph with at most \( \varepsilon e[U]^2/\sqrt{\log n} \) edges from \( E_R \). Then it finds a balanced directed cut in this graph using the algorithm of Agarwal, Charikar, Makarychev, and Makarychev [ACMM05].

The SDP that we solve in each iteration (given in Section 4.2) defines a "directed distance" on the vertices. We show that the edges from \( E_R \) get successively "sparsified" by a constant factor in the subgraphs \( H_t \) as mentioned above, that follows from Theorem 4.6 (Structural Theorem): after the preprocessing step, most of the edges from \( E_R \) are "longer" (according to this directed distance) than a certain threshold \( \delta_t = 2^{-t} \) (at most \( O(\delta^t) \) fraction of the edges are longer than \( \delta_t \)). However, the use of the directed distance (as opposed to an undirected distance in [MMV12]) introduces some difficulties, which we handle by performing a preprocessing step (see Lemma 4.3) at the start of each iteration. This ensures that the directed distance from the SDP at step \( t \) can be well approximated by an undirected distance (up to an additive error of \( \delta_t \)) in the remaining graph. Now, we can use the ideas from [MMV12] to identify the vertices to remove, and edges to cut to obtain \( H_{t+1} \).

4.2 SDP relaxation

We now describe the SDP relaxation for Minimum Directed Balanced Cut. Our algorithm will iteratively construct a sequence of subgraphs \( H = H_0 \supset H_1 \supset H_2 \supset \ldots \). The SDP will be a relaxation for the following problem: find a bisection \((S, T)\) of \( U \) that minimizes the number of cut edges in a given set \( E' \subset E_H \). In iteration \( i \), \( E' \) will be the set of edges in the graph \( H_i \).

We have a unit vector variable \( \vec{u} \) for every vertex \( u \in U \). We also have a special unit vector \( \vec{v}_0 \). Denote by \( d(u,v) \) the directed distance \((\text{ACMM05})\):

\[
d(u, v) = \| \vec{u} - \vec{v} \|^2 + 2\langle \vec{v}_0, \vec{u} - \vec{v} \rangle.
\]

We write an SDP relaxation as follows. subject to: for every \( u, v, w \in U \),

\[
\frac{1}{2} \sum_{u,v \in U} \| \vec{u} - \vec{v} \|^2 \geq |U|^2 - 1
\]

and

\[
\| \vec{u} - \vec{v} \|^2 + \| \vec{v} - \vec{w} \|^2 \geq \| \vec{u} - \vec{w} \|^2
\]

\[
d(u,v) \geq 0
\]

We denote this SDP relaxation by \( \text{SDP}(U, E') \). We denote the value of the optimal SDP solution by \( \text{sdp-cost}(U, E') \). The constraints of the SDP define an undirected distance between the vertices (given by \( |\vec{u} - \vec{v}|^2 \)), that is referred to as the \( \ell_2^2 \) metric. Further, \( d(u,v) \) defines a directed distance (directed semi-metric, to be precise) between the vertices (see Lemma 4.2 for details).

Suppose that \((S, T)\) is a bisection of \( U \). Then in the intended solution, we assign \( \vec{u} = \vec{v}_0 \) if \( u \in S \) and \( \vec{u} = -\vec{v}_0 \) if \( u \in T \). Note that only edges \((u,v)\) that go from \( S \) to \( T \) contribute to the objective function. And for every such edge \((u, v)\), we have \( \frac{1}{2}d(u,v) = 1 \). Thus the objective function equals \( \text{cost}_{\ell_2^2}(S, T) \).

Definition 4.1 Given an SDP solution on \( U \), we define directed balls \( \text{Ball}^+(u, r) \) and \( \text{Ball}^-(u, r) \) as follows

\[
\text{Ball}^+(u, r) = \{ v : d(u, v) \leq r \}
\]

and

\[
\text{Ball}^-(u, r) = \{ v : d(u, v) \leq r \}.
\]

Let \( \text{Ball}^0(u, r) = \{ v : \| \vec{u} - \vec{v} \|^2 \leq r \} \). The width of a subset \( M \subset U \) is \( \text{width}(M) = \max_{u,v \in M} (\langle \vec{v}_0, \vec{u} - \vec{v} \rangle) \). The \( \ell_2^2 \)-diameter of a set \( M \) is \( \text{diam}(M) = \max_{u,v \in M} (\| \vec{u} - \vec{v} \|^2) \).

Lemma 4.2 Consider the directed distance function \( d(u,v) \) defined by a feasible solution to \( \text{SDP}(U, E') \).

1. The function \( d(u,v) \) defines a directed semi-metric on \( U \). That is, \( d(u,v) \geq 0 \) and \( d(u,v) + d(v,w) \geq d(u,w) \).

2. For every \( u \) and \( v \) in \( U \),

\[
\text{Ball}^0(u, r/2) \cap M \subset \text{Ball}^+(u, r) \cap M
\]

and

\[
\text{Ball}^0(u, r/2) \cap M \subset \text{Ball}^-(u, r) \cap M
\]

3. Suppose that \( M \) is a subset of \( U \) of width \( w \) then for every \( u, v \in M \), we have

\[
\| \vec{u} - \vec{v} \|^2 \leq 2d(u, v) \leq \| \vec{u} - \vec{v} \|^2 + 2w.
\]

4. If \( \text{width}(M) \leq w \) (as in part 3), then for \( u \in M \),

\[
\text{Ball}^0(u, r/2) \cap M \subset \text{Ball}^+(u, r) \cap M
\]

and

\[
\text{Ball}^0(u, r/2) \cap M \subset \text{Ball}^-(u, r) \cap M
\]

\subset \text{Ball}^0(u, r + 2w) \cap M.
4.3 Preprocessing

The directed semi-metric $d(u, v)$ defined by a feasible SDP solution has two components: the undirected $L_2^2$ component $\|\bar{u} - \bar{v}\|^2$ and the projection of $u - v$ on to $\bar{v}_0$. Here, we show how to preprocess the graph, so that in the remaining graph, the directed semi-metric $d(u, v)$ can be approximated well by the undirected $L_2^2$ distance. The preprocessing algorithm is motivated by the following observation: edges $(u, v)$ whose distance $d(u, v)$ has a large contribution from the projection $(\bar{u}_0, \bar{v})$, can be separated cheaply by looking at their projections on $\bar{v}_0$. We now describe the preprocessing step, along with its guarantees.

**Lemma 4.3** There is an algorithm that given a $H'(U', E')$, a parameter $\delta > 0$ and an SDP solution $\{\bar{u}\}$ of cost sdpcost does the following. Either it finds a 0.1-balanced cut in $H'$ of cost $O(\text{sdpcost}/\delta)$, or it partitions $U'$ into sets $L, C$, and $R$ such that

1. each of the sets $L$ and $R$ contains at most $|U'|/10$ vertices;
2. the total cost of directed cuts $(R, C)$, $(C, L)$ and $(R, L)$ is $O(\text{sdpcost}/\delta)$;
3. width($C$) $\leq \delta/8$.

**Proof.** We run the following algorithm.

**Preprocessing Algorithm**

**Input:** Graph $H(U', E')$, parameter $\delta > 0$ and a feasible SDP solution $\{\bar{u}\}$, as in Lemma 4.3.

**Output:** A directed balanced cut or a partition of $U'$ into three sets $L$, $C$, and $R$.

1. Let $\mu$ be the median of the set $\{\langle \bar{v}_0, \bar{u} \rangle : u \in U\}$.
2. Let $L_r = \{u \in U : \langle \bar{v}_0, \bar{u} \rangle < \mu - \tau\}$, $C_r = \{u \in U : \langle \bar{v}_0, \bar{u} \rangle < \mu + \tau\}$, and $R_r = \{u \in U : \langle \bar{v}_0, \bar{u} \rangle > \mu + \tau\}$.
3. Choose $\tau \in (0, \delta/16)$ that minimizes the total cost of directed cuts $(R_r, C_r)$, $(C_r, L_r)$, and $(R_r, L_r)$.
4. If $|L_r| > |U'|/10$ return the cut $(C_r \cup R_r, L_r)$; if $|R_r| > |U'|/10$ return the cut $(R_r, C_r \cup L_r)$.
5. Otherwise, return sets $L = L_r$, $C = C_r$, and $R = R_r$.

Note that if the algorithm outputs a directed cut, the side of the cut that equals $L_r$ or $R_r$ contains at least $|U'|/10$ vertices, and the other side, which equals $L_r \cup C_r$ or $R_r \cup C_r$, contains at least $|U'|/2$ vertices since $|L_r \cup C_r| \geq |u \in U : \langle \bar{v}_0, \bar{u} \rangle \leq \mu| \geq |U'|/2$ (by the definition of $\mu$); similarly, $|R_r \cup C_r| \geq |U'|/2$. Therefore, the cut is 0.1-balanced. If the algorithm outputs sets $L, C$, and $R$ then condition (1) clearly holds.

Now we will prove that if the algorithm outputs a directed cut then its cost is at most $32 \text{sdpcost}/\delta$, and if the algorithm outputs sets $L, C$, and $R$ then condition (2) holds. To this end, we show that the total cost of cuts $(R_r, C_r)$, $(C_r, L_r)$, and $(R_r, L_r)$ is at most $32 \text{sdpcost}/\delta$. Suppose that we choose $\tau$ uniformly at random from $(0, \delta/16)$. Consider an edge $(u, v) \in E'$. We compute the probability that $(u, v)$ is cut by one of the directed cuts $(C_r, L_r)$ and $(R_r, L_r)$. If $\langle \bar{v}_0, \bar{u} \rangle \leq \langle \bar{v}_0, \bar{v} \rangle$ then the edge $(u, v)$ cannot be cut. If $\langle \bar{v}_0, \bar{u} \rangle \geq \langle \bar{v}_0, \bar{v} \rangle$ then the probability that $(u, v)$ is cut is at most $\langle \bar{v}_0, \bar{u} - \bar{v} \rangle/\delta(\delta/16) \leq 4d(u, v)/\delta$ (by Lemma 4.2). Since $\sum_{(u, v) \in E} d(u, v) \leq \text{sdpcost}$, the expected total number of cut edges is at most $32 \text{sdpcost}/\delta$. Thus for the optimal value of $\tau \in (0, \delta)$, the total number of cut edges is also at most $32 \text{sdpcost}/\delta$.

Finally, width($C$) $= \max_{u \in U}(\langle \bar{v}_0, \bar{u} \rangle - \langle \bar{v}_0, \bar{v} \rangle) \leq (\mu + \delta/16) - (\mu - \delta/16) = \delta/8$.}

4.4 Heavy Vertices and Geometric Expansion

In this subsection, we first remind the definitions of heavy vertices and geometric expansion from [MMV12] and state some related results from [MMV12]. Then we describe the heavy vertex removal procedure for directed graphs (which is a modification of the vertex removal procedure for undirected graphs from [MMV12]).

**Definition 4.4** Let $U$ be a set of vertices, and $M \subseteq U$. Suppose that $\{\bar{u}\}$ is a feasible SDP solution. We say that a vertex $u \in M$ is heavy in $M$ if $|\text{Ball}^0(u, \delta) \cap M| \geq \delta^2|U|$. We denote the set of all heavy vertices by $\mathcal{H}_M(M)$. (Note that the set of heavy vertices depends on the SDP solution.)

**Definition 4.5** (Geometric Expansion) A graph $H = (U, E_H)$ satisfies the geometric expansion property with cut value $X$ at scale $\delta$ if for every feasible SDP solution $\{\bar{u}\}$ and every subset of vertices $M \subseteq U$ satisfying $\mathcal{H}_M(M) = \emptyset$ w.r.t $\{\bar{u}\}$,

$$|\langle u, v \rangle \in E \cap (M \times M) : \|\bar{u} - \bar{v}\|^2 \leq \delta/2| | \leq 2\delta^2 X.$$

A graph $H = (U, E_H)$ satisfies the geometric expansion property with cut value $X$ up to scale $\delta^*$ if it satisfies the geometric expansion property for every $\delta \in [\delta^*, 1]\cap\{2^{-t} : t \in \mathbb{N}_{\geq 0}\}$.

We will also use the following theorem, which is a simple variant of Theorem 5.1 proved in [MMV12]. (We omit the proof of Theorem 4.6 in this paper.)

**Theorem 4.6** Let $V$ be a set of vertices of size $n$, $\varepsilon \in (0, 1)$, and $D > 2$. Let $E_R$ be an $\varepsilon$-random set of edges. Then with probability $1 - 2^{-\Theta(n)}$ the random set $\text{SRD}(V, E_R)$ satisfies the following property: every graph $H = (U, E_H) \in \text{SRD}(V, E_R)$ is geometrically expanding with cut cost $X = C \max\{\varepsilon^2|U|, nD(\log^2 D)\}$ up to scale $1/\sqrt{D}$.

Now we present a procedure that removes heavy vertices from a graph. This procedure is similar to the Heavy Vertex Removal Lemma from [MMV12].
Lemma 4.7 There exists a polynomial-time algorithm that given sets $U' \subset U$, a feasible SDP solution $\{\tilde{u}\}$, and a subset $M \subseteq U'$ of width $w \leq \delta/4$, finds a set of vertices $M' \subseteq M$ and a partition of $M \setminus M'$ into disjoint sets $Z_1, \ldots, Z_k$ (for some $k \geq 0$) such that

- the set $M'$ does not contain any $\delta$-heavy vertices (i.e., $H_{\delta}(M') = \emptyset$) w.r.t. the SDP solution $\{\tilde{u}\}$,
- $|Z_i| \leq \delta/4 |U|$ for each set $Z_i$,
- for every two vertices $u^*$ and $v^*$ in $M$, we have
  $$\Pr(\exists u \in Z_i \text{ s.t. } u^* \in Z_i \text{ and } v^* \notin Z_i) \leq \frac{C(\delta^{-1} + \delta^{-2} \frac{|M \setminus M'|}{|U|})d(u^*, v^*)}{128},$$

Proof. We use the following algorithm. If $\delta \geq 1/128$, we run the algorithm with $\delta' = 1/128$.

Heavy Vertices Removal Procedure

Input: a set of vertices $U$, a subset $M \subseteq U$, an SDP solution $\{\tilde{u}\}$, a parameter $\delta \in (0, 1/128]$.

Output: a set $M' \subseteq M$, partition $M \setminus M' = Z_1 \cup \cdots Z_k$ (for some $k \geq 0$);

- while ($H_{\delta'}(M) \neq \emptyset$)
  - Connect heavy vertices in $M$ at $\ell_2^2$ distance at most $8\delta$ with an undirected edge and denote the new set of undirected edges by $E'$; that is, $A = \{(u, v) \in H_{\delta'}(M) \times H_{\delta'}(M) : \|u - v\| \leq 8\delta\}$.
  - Break undirected graph $(H_{\delta'}(M), A)$ into connected components.
  - Pick a random $r \in [2\delta, 3\delta]$.
  - Remove components of small diameter: For each connected component $Q$ with
    $$\text{diam}(Q) = \max_{u, v \in Q} \|u - v\| \leq 1/8,$$
    let $B_Q = \bigcup_{u \in Q} \text{Ball}^+(u, r) \cap M$
    $$= \{v \in M : \exists u \in Q \text{ s.t. } d(u, v) \leq r\}.$$
    Denote the set of all connected components of diameter at most 1/8 by $Q$.
  - Remove a maximal independent set: In the remaining set $H_{\delta'}(M) \setminus \bigcup_{Q \in Q} Q$ find a maximal independent set $I$ (this is done independently of the random variable $r$, e.g., using a deterministic greedy algorithm). For each $u \in I$, let $B_u = \text{Ball}^+(u, r) \cap M$.
    - Remove sets $B_Q$ and $B_u$ from $M$: $M' = M \setminus \left( \bigcup_{Q \in Q} B_Q \cup \bigcup_{u \in I} B_u \right)$;

- Let sets $Z_1, \ldots, Z_k$ be the sets that we removed from $M$ (in the order in which we removed them).
- return $M' = M$ and sets $\{Z_i\}$.

Analysis. It is clear that the algorithm always terminates in polynomial-time (since at every step at least one vertex is removed). When the algorithm terminates $H_{\delta'}(M) = \emptyset$ by the condition of the "while" loop. In order to bound the size of sets $Z_i$, we prove that the $\ell_2^2$-diameter of every set $Z_i$ is at most 1/4. Each set $Z_i$ is either equal to some $B_u$ or some $B_Q$. Every set $B_u$ removed from $M$ at one of the iterations is contained in $\text{Ball}^+(u, 3\delta) \cap M \subseteq \text{Ball}^+(u, 3\delta + 4\delta) \cap M$ by Lemma 4.2; thus its $\ell_2^2$-diameter is at most $6\delta + 4\delta \leq 1/4$. Every set $B_Q$ is contained in $\bigcup_{u \in Q} \text{Ball}^+(u, 3\delta) \cap M \subseteq \bigcup_{u \in Q} \text{Ball}^+(u, 3\delta + 4\delta) \cap M$. Since the $\ell_2^2$-diameter of $Q$ is at most $1/8$, the $\ell_2^2$-diameter of $Q$ is at most $1/8 + 6\delta + 4\delta < 1/4$. We get that for every $Z_i$:

$$\sum_{u, v \in U} \|u - v\|^2 \leq (|U|^2 - |Z_i|^2) \cdot \max_{u, v \in U} \|u - v\|^2.$$

By the SDP spreading constraint, the left hand side is greater than or equal to 2($|U|^2 - 1$), thus $|Z_i|^2 \leq (8|U|^2 + 8)/15$, and $|Z_i| \leq \frac{3}{4} |U|$ (when $|U| \geq 8\delta$). Now we verify the third item of the lemma. Fix two vertices $u^*$ and $v^*$ and consider one iteration of the algorithm. We may assume that the algorithm first picks the independent set $I$ and a collection of connected components $Q$, and only then chooses random $r \in [2\delta, 3\delta]$. Observe, that the $\ell_2^2$-distance between two arbitrary vertices $u$ and $v$ (in $I$) is at least $8\delta$ (because $I$ is an independent set), the $\ell_2^2$-distance between every two sets in $Q$ is at least $8\delta$ (otherwise, these two sets would form one connected component), and the $\ell_2^2$-distance between every $Q \in Q$ and $u \in I$ is at least $8\delta$ (again because $Q$ is a connected component, and $u \notin Q$). Thus, $u^*$ may belong to at most one $\bigcup_{Q \in Q} \text{Ball}^+(u, 3\delta)$ or $\text{Ball}^+(u, 4\delta)$. By Lemma 4.2, $\text{Ball}^+(u, 3\delta) \subseteq \text{Ball}^+(u, 4\delta)$ (since $w < \delta/2$), thus $u^*$ may belong to at most one $\bigcup_{Q \in Q} \text{Ball}^+(u, 3\delta)$ or $\text{Ball}^+(u, 3\delta)$. If $u^* \in \text{Ball}^+(u, 3\delta) \cap M$ and $v^* \in M$, then

$$\Pr(u^* \in \text{Ball}^+(u, r), v^* \notin \text{Ball}^+(u, r)) \leq \Pr(d(u, u^*) \leq r \leq d(u, v^*)) \leq \frac{d(u^*, v^*)}{3\delta}.$$
Thus, \( E[T] \leq 1 + O(E[M \setminus M/(δ(U))] \), and the third item of Lemma 4.7 follows. Observe, that if \( I = \emptyset \), then the algorithm terminates. If \( I \neq \emptyset \), there exists at least one connected component \( L \) with \( diam(L) \geq 1/8 \). The maximal independent set in \( L \) must contain at least \( Ω(δ^{-1}) \) vertices, since for every edge \( (u, v) \in A, |\bar{u} - \bar{v}| \leq 8^i \). Thus, \( |I| \geq Ω(δ^{-1}) \). Since each \( u \in I \) is \( δ \)-heavy and \( r > 2δ, |B_u| \geq |B(u, 2δ)\cap M| \geq |B(u, δ)\cap M| \geq δ^2/|U| \). Hence (using the fact that sets \( B_u \) are disjoint),

\[
\bigcup_{u \in I} B_u = \sum_{u \in S} |B_u| \geq Ω(δ|U|).
\]

Now we combine Lemma 4.7 and Lemma 4.3 and obtain the following lemma.

**Lemma 4.8** There is a polynomial-time algorithm that given a graph \( H(U, E_H) \), its subgraph \( H'(U', E') \) and a feasible SDP solution \( \{\bar{u}\} \) of cost at most \( \text{sdp-cost}(U, E') \), partition \( U' \) into three sets \( S', T' \) and \( U'' \) satisfying the following conditions:

- The set \( U'' \) does not contain any \( δ \)-heavy vertices w.r.t. \( \{\bar{u}\} \) (H\(_G(U'') = \emptyset \).
- \( |S'| \leq 0.9|U| \) and \( |T'| \leq 0.9|U| \).
- The total cost of directed cuts \( (S', U''), (S', T') \) and \( (U'', T') \) is at most \( C(δ^{-1} + δ^{-2} \frac{E(U \setminus U'')}{|U|}) \cdot \text{sdp-cost} \).

**Proof.** We run the algorithm from Lemma 4.3. If the algorithm returns a directed balanced cut in \( (U, E') \) of cost \( O(\text{sdp-cost} / δ) \), we let \( S', T' \) be this cut, and let \( U'' = \emptyset \). The first condition trivially holds since \( U'' = \emptyset \). The second and third conditions hold by Lemma 4.3. Otherwise, the algorithm returns three sets \( L, C \) and \( R \). We run the algorithm from Lemma 4.7 on \( M = C \) and get a partition of \( M \) into a set \( M' \) and a family of sets \( Z_1, \ldots, Z_k \).

We consider two cases. First, assume that \( |R \cup Z_1 \cup \ldots \cup Z_k| \leq 0.9|U| \). Then we let \( S' = R \cup Z_1 \cup \ldots \cup Z_k \), \( T' = L \) and \( U'' = M' \). The first property holds by Lemma 4.7. By our assumption, \( |S'| \leq 0.9|U| \) by Lemma 4.3, \( |T'| = |L'| \leq 0.1|U'| \leq 0.1|U| < 0.9|U| \). It follows from Lemmas 4.3 and 4.7 that the total cost of cuts \( (S', U''), (S', T') \) and \( (U'', T') \) is \( O\left(δ^{-1} + δ^{-2} \frac{E(U \setminus U'')}{|U|}\right) \cdot \text{sdp-cost} \).

Now let us assume \( |R \cup Z_1 \cup \ldots \cup Z_k| > 0.9|U| \). We find the last index \( i \) such that \( |R \cup Z_1 \cup \ldots \cup Z_i| \leq 0.9|U| \). Let \( S' = R \cup Z_1 \cup \ldots \cup Z_i \), \( T' = C \cup U \setminus Z_{i+1} \cup \ldots \cup Z_k \) and \( U'' = \emptyset \). The first property trivially holds by Lemma 4.7. Verify that the second property holds. By our choice of \( i \), \( |S'| \leq 0.9|U| \). Since \( |S' \cup Z_{i+1}| > 0.9|U| \) and \( |Z_{i+1}| < 0.75|U| \), we have \( |S'| > 0.15|U| \). and therefore \( |T'| = |U'| - |S'| < 0.85|U| \). Finally, it follows from Lemmas 4.3 and 4.7 that the cost of the cut \( (S', T') \) is at most \( O\left(δ^{-1} + δ^{-2} \frac{E(U \setminus U'')}{|U|}\right) \cdot \text{sdp-cost} \).

4.5 Algorithm for Minimum Directed Balanced Cut

Now we are ready to present our algorithm.

**Input:** a graph \( H(U, E_H) \); **Output:** a directed balanced cut \( (S', T') \);

- Let \( t = 0 \), \( H_0 = H, U_0 = U, E_0 = E_H \).
- while \( (δ_t \geq (log n)^{-1/4}) \) do
  - A. Let \( δ = δ_t = 2^{-t} \).
  - B. Solve the SDP relaxation \( \text{SDP}(U, E_t) \) for the Minimum Directed Balanced Cut.
  - C. Run the algorithm from Lemma 4.8, and obtain sets \( S_{t+1} = S', T_{t+1} = T' \) and \( U_{t+1} = U'' \).
  - D. Consider the induced graph \( H_t(U_{t+1}) \). Remove all edges \( (u, v) \in H_t(U_{t+1}) \) with \( d(u, v) \geq δ_t/4 \). Denote the obtained graph by \( H_{t+1} = (U_{t+1}, E_{t+1}) \).
  - E. Let \( t = t + 1 \).

- Find a directed 0.1-balanced cut \( (S'', T'') \) in the graph \( (U, E_t) \) using the algorithm of Agarwal, Charikar, Makarychev and Makarychev [ACMM05].
- Let \( A_1 = S_1, \ldots, A_{t-1} = S_t, A_{t+1} = S'' \cap U_t, A_{t+2} = T'' \cap U_t, A_{t+3} = T_{t+1}, \ldots, A_{2t+2} = T_t \).
- Find the last \( t' \) s.t. \( \sum_{i=t+1}^{t'} A_i < 0.95|U| \).
- Let \( S' = \bigcup_{i=t+1}^{t'} A_i \) and \( T' = \bigcup_{i=t+1}^{t'} A_i \).
- return \( (S', T') \).

**Analysis.** By Theorem 4.6, graph \( (U, E_R \cap (U \cup U')) \) is geometrically expanding with cut value

\[ X = C \max\{\varepsilon^2 |U|, n \sqrt{\log n (\log^2 \log n)}\} \]

up to scale \( (\log n)^{-1/4} \). Consider iteration \( t \) of the algorithm. By Lemma 4.8, the set \( U_{t+1} \) contains no \( δ_t \)-heavy vertices. Therefore, by the definition of geometric expansion, there are at most \( 2δ^2_t X \) edges \( (u, v) \in E_H \cap (U' \cup U'') \) with \( |u - v| \leq δ_t/2 \). Note that if \( |u - v| \leq δ_t/2 \) then \( d(u, v) \geq |u - v| - 2\text{width}(U'') \geq δ_t/4 \).

Thus at step \( D \), we remove all but at most \( 2δ^2_t X \) random edges from \( H_t \). That is, \( |E_t \cap E_{t+1}| \leq δ^2_t X \). Therefore, \( \text{sdp-cost}(U, E_{t+1}) \leq 2δ^2_t X \).

Now we bound the cost of the cut. Note that the total number of edges cut at step \( C \) — edges that go from \( S_{t+1} \) and \( T_{t+1} \), from \( S_{t+1} \) to \( U_{t+1} \), or from \( U_{t+1} \) to \( T_{t+1} \) — is at most \( \sum_{i=1}^{t} |E(H_t, U) \cap U_{t+1}| \cdot X \). Since \( \sum_{i=1}^{t} δ_t < 2, \) and \( \sum_{i=1}^{t} |E(H_t, U) \cap U_{t+1}|/|U| < 1, \) we get that the total number of edges cut at step \( C \) in all iterations is \( O(X) \). We cut at most \( \text{sdp-cost}(U, E_t)/(δ_t/4) = O(δ_t X) \) edges at step \( D \) in one iteration; and we cut at most \( O(X) \) edges in total all iterations. Finally, we bound the cost of \( (S'', T'') \). The planted cut \( (S, T) \) in \( U \) cuts at most \( |E_t \cap E_R| \leq δ^2_t X \) edges in \( E_t \). Thus (here, \( δ_t < (\log n)^{-1/4} \))

\[ \text{cost}_{E_t}(S'', T'') \leq O(\sqrt{\log n}) \cdot \text{cost}_{E_t}(S, T) \leq O(\sqrt{\log n})δ^2_t X = O(X). \]
Finally, we verify that the cut $(S', T')$ is balanced. By our choice of $j^*$, $|S'| < 0.95|U|$. Since $|A_i| \leq 0.9|U|$ for all $i$,
\[
|T'| = |U| - |S'| = |U| - \left( \sum_{i=1}^{j^*+1} |A_i| \right) + |A_{j^*+1}|
\leq |U| - 0.95|U| + 0.9|U| = 0.95|U|.
\]

\[\square\]

5. REFERENCES


[Ka92] V. Kann. *On the Approximability of NP-complete Optimization Problems*. PhD


